

**CLAIMS**

What is claimed is:

- 1 1. A computer-implemented method for dynamic instrumentation of an executable  
2 application program using an instrumentation program, the application program  
3 including a plurality of original functions, each original function having an entry point  
4 and an endpoint, comprising:  
5       creating a shared memory segment for the instrumentation program and the  
6 application program;  
7       upon initial invocation of the original functions in the application program,  
8 creating in the shared memory segment corresponding substitute functions including  
9 instrumentation code; and  
10       executing the substitute functions in lieu of the original functions in the  
11 application program.
- 1 2. The method of claim 1, further comprising:  
2       patching the function entry points with breakpoint instructions; and  
3       creating the substitute functions upon encountering the breakpoint instructions.
- 1 3. The method of claim 2, further comprising replacing the break instruction at the  
2 entry points of the functions in the application program with branch instructions that  
3 target the substitute functions.
- 1 4. The method of claim 3, wherein the executable application program includes  
2 one or more branch instructions having target addresses that reference entry points of  
3 one or more of the original functions, further comprising:  
4       after creating a substitute function corresponding to an original function, for a  
5 branch instruction that references the original function replacing the target addresses to  
6 reference the substitute function.
- 1 5. The method of claim 1, wherein the executable application program includes  
2 one or more branch instructions having target addresses that reference entry points of  
3 one or more of the original functions, further comprising:

4 after creating a substitute function corresponding to an original function, for a  
5 branch instruction that references the original function replacing the target addresses to  
6 reference the substitute function.

1 6. The method of claim 1, further comprising:  
2 copying a segment of the executable application program to selected area of  
3 memory by the instrumentation program;  
4 replacing the segment of the application program with code that allocates the  
5 shared memory by the instrumentation program;  
6 executing the code in the application program that allocates the shared memory  
7 segment; and  
8 restoring the segment of the executable application from the selected area of  
9 memory to the application program by the instrumentation program after the shared  
10 memory is allocated.

1 7. The method of claim 6, further comprising:  
2 patching the function entry points with breakpoint instructions; and  
3 creating the substitute functions upon encountering the breakpoint instructions.

1 8. The method of claim 7, further comprising replacing the break instruction at the  
2 entry points of the functions in the application program with branch instructions that  
3 target the substitute functions.

1 9. The method of claim 8, wherein the executable application program includes  
2 one or more branch instructions having target addresses that reference entry points of  
3 one or more of the original functions, further comprising:  
4 after creating a substitute function corresponding to an original function, for a  
5 branch instruction that references the original function replacing the target addresses to  
6 reference the substitute function.

1 10. The method of claim 6, wherein the executable application program includes  
2 one or more branch instructions having target addresses that reference entry points of  
3 one or more of the original functions, further comprising:

4 after creating a substitute function corresponding to an original function, for a  
5 branch instruction that references the original function replacing the target addresses to  
6 reference the substitute function.

1 11. The method of claim 6, wherein the executable application program includes a  
2 plurality of threads and further comprising:  
3 before the step of copying the segment of the executable application program  
4 suspending all threads of the executable application program, and selecting one of the  
5 suspended threads; and  
6 after replacing the segment of the executable application program with the code  
7 that allocates the shared memory, resuming execution of the one of the suspended  
8 threads at the code that allocates the shared memory.

1 12. The method of claim 11, further comprising:  
2 patching the function entry points with breakpoint instructions; and  
3 creating the substitute functions upon encountering the breakpoint instructions.

1 13. The method of claim 12, further comprising replacing the break instruction at  
2 the entry points of the functions in the application program with branch instructions  
3 that target the substitute functions.

1 14. The method of claim 13, wherein the executable application program includes  
2 one or more branch instructions having target addresses that reference entry points of  
3 one or more of the original functions, further comprising:  
4 after creating a substitute function corresponding to an original function, for a  
5 branch instruction that references the original function replacing the target addresses to  
6 reference the substitute function.

1 15. An apparatus for dynamic instrumentation of an executable application program  
2 by an instrumentation program, the application program including a plurality of original  
3 functions, each original function having an entry point and an endpoint, comprising:  
4 means for creating a shared memory segment for the instrumentation program  
5 and the application program;

*Sub*

add 1

[illegible]